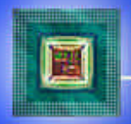
Next Generation Network Processor Architecture

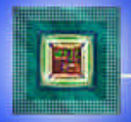
Matthew Adiletta
Intel Fellow, Director of Communications Processor Architecture

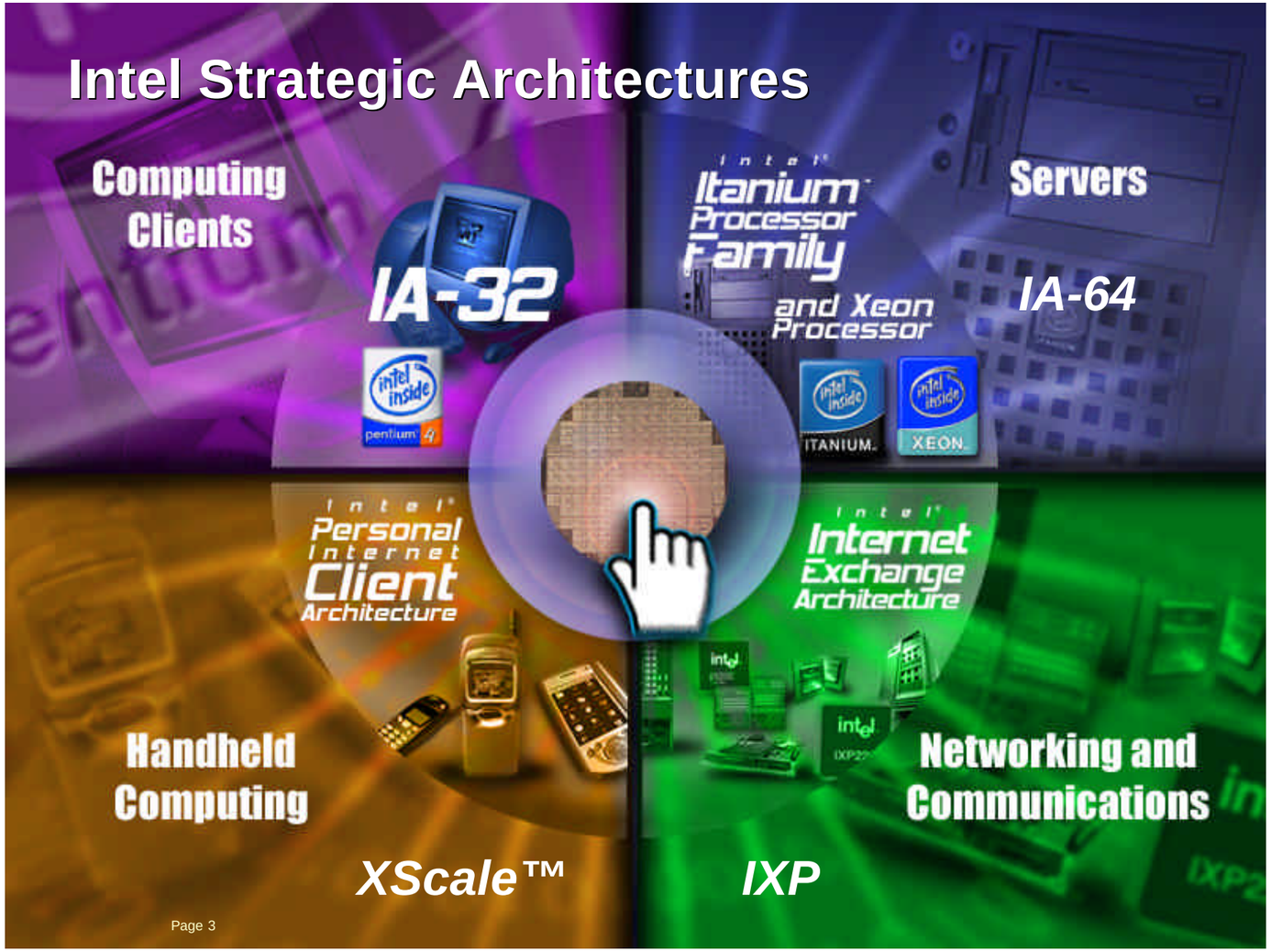
October 18, 2001



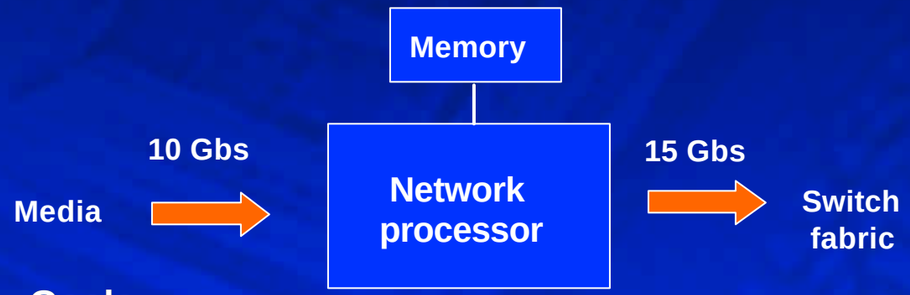
Agenda

- Introduction
- Intel technology drives performance
- Network processing challenges at 10 Gbs & beyond
- Next generation micro architecture
- Software pipelining delivers rich product features
- Demonstration of software pipelining at 10 Gbs
- Summary




Intel Strategic Architectures
Computing Clients
IA-32
Intel Itanium Processor Family and Xeon Processor
Servers
IA-64
Intel Personal Internet Client Architecture
Handheld Computing
XScale™
Intel Internet Exchange Architecture
Networking and Communications
IXP

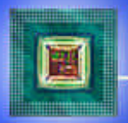
Single chip 10 Gbs network processor



- Goals
  - Deliver wire speed store and forward processing at 10 Gbs and beyond
  - Provide fully programmable processing
  - Integrate forwarding and traffic management
- Benefits
  - Supports broad range of applications
  - Enables developer differentiation
  - Lowers system cost, board space and power

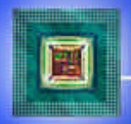
Intel's next generation network processor technology

- The Intel OC-192 network processor employs custom circuit design and layout to achieve 1.4GHz operation - well beyond ASIC capability
  - Provides performance with flexibility
- The Intel OC-192 network processor employs leading edge process technology: .13u
  - Provides network intelligence at a low cost
- The Intel OC-192 network processor utilizes IA-32 and IA-64 design methodology and process technology
  - Proven sustainable performance roadmap
- Intel will be on the leading edge of silicon technology and process improvements



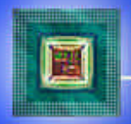
Network processing challenges at 10 Gbs & beyond

- Store and forward cell / packet processing have two characteristic network processing problems which require solutions:
  - The dependence problem
    - Cells / packets dependent upon predecessor cells / packets
    - Examples: CRC calculations on distinct cells for an overall packet; sequence management
  - The independence problem
    - Common data-structure accesses by non-related cells / packets
    - Example: enqueuing and dequeueing to transmit queues



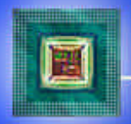
The dependence problem:

- If current cell / packet (C/P) is dependent on previous cells or packets then ordering of processing is critical:
  - CASE 1: Time-wise distant relationships require external buffering of context. Current C/P must “fetch” context from memory;
  - CASE 2: Time-wise relatively close relationships require “mutex” scheme to eliminate race of previous C/P context write update in-flight to memory vs. current C/P reading memory context;
  - CASE 3: Time-wise very close relationships may need special schemes to maintain line-rate if there is insufficient time to allow write of context to memory then next C/P read from memory of that context.



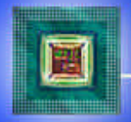
The independence problem

- If the line rate is sufficiently fast then performing cell / packet relationship independent functions may require special schemes:
  - "Allocation" of buffers from a linked list (LL) requires traversing links implemented as dependent reads
  - "Freeing" of buffers onto a LL requires LL traversal
  - If transmit queues in memory are maintained as linked list structures, then back-to-back LL updates may require special schemes:
    - Enqueue A then dequeue A
    - En[De]Q-X then En[De]Q-Y from the same queue
  - Many transmit scheduler algorithms employ LL's which require line rate traversal.



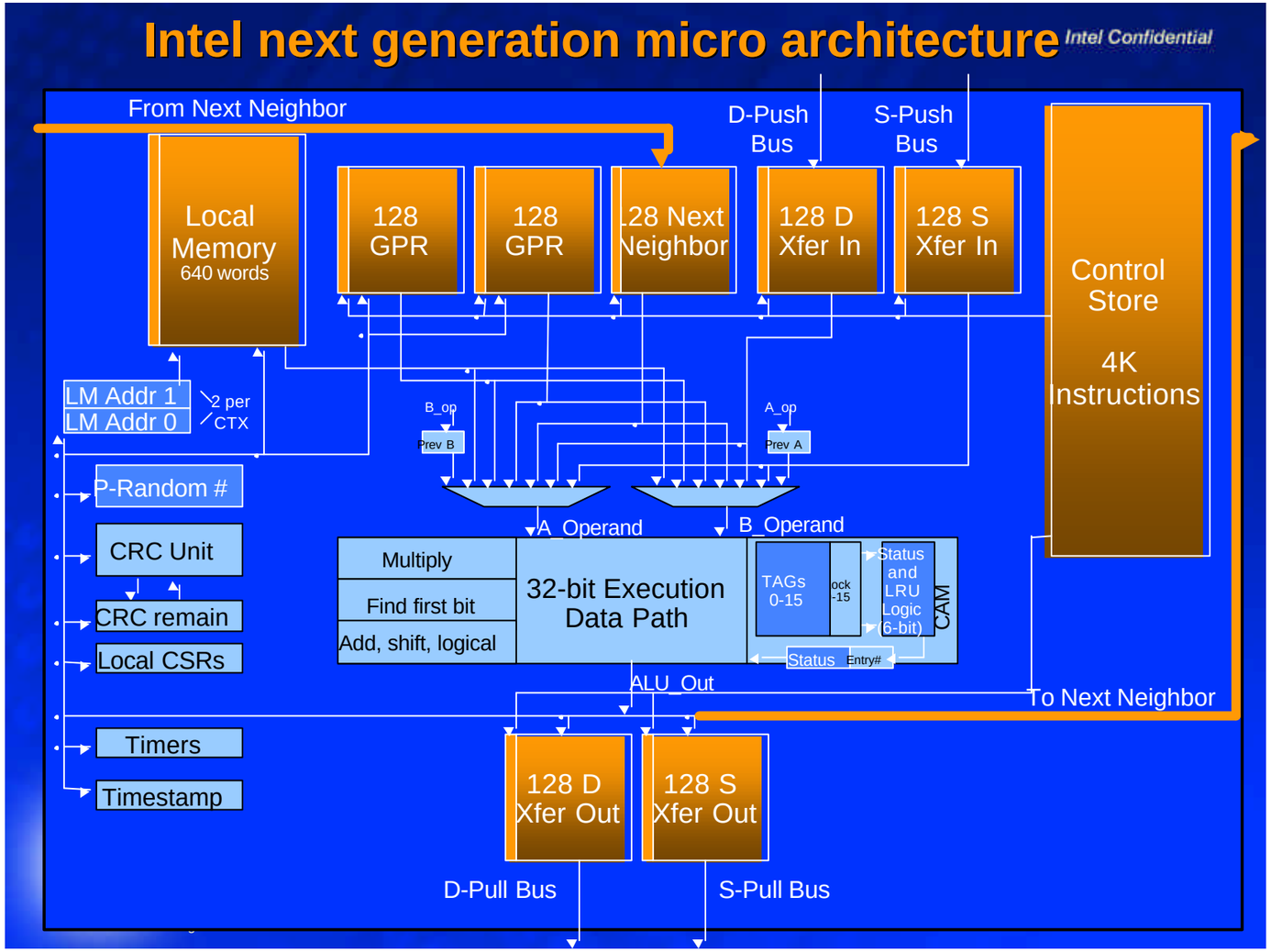
Potential line rate solutions:

- Employ very low latency hardware state machines (ASICs)
  - Lack of flexibility
  - Algorithm changes require silicon spin
  - Verification of all corner cases (time to market, development cost)
  - Does the solution fit the customer's problem?
- Employ multi-processing and pipelining
  - But how?



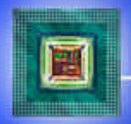
Intel next generation micro architecture

Intel Confidential



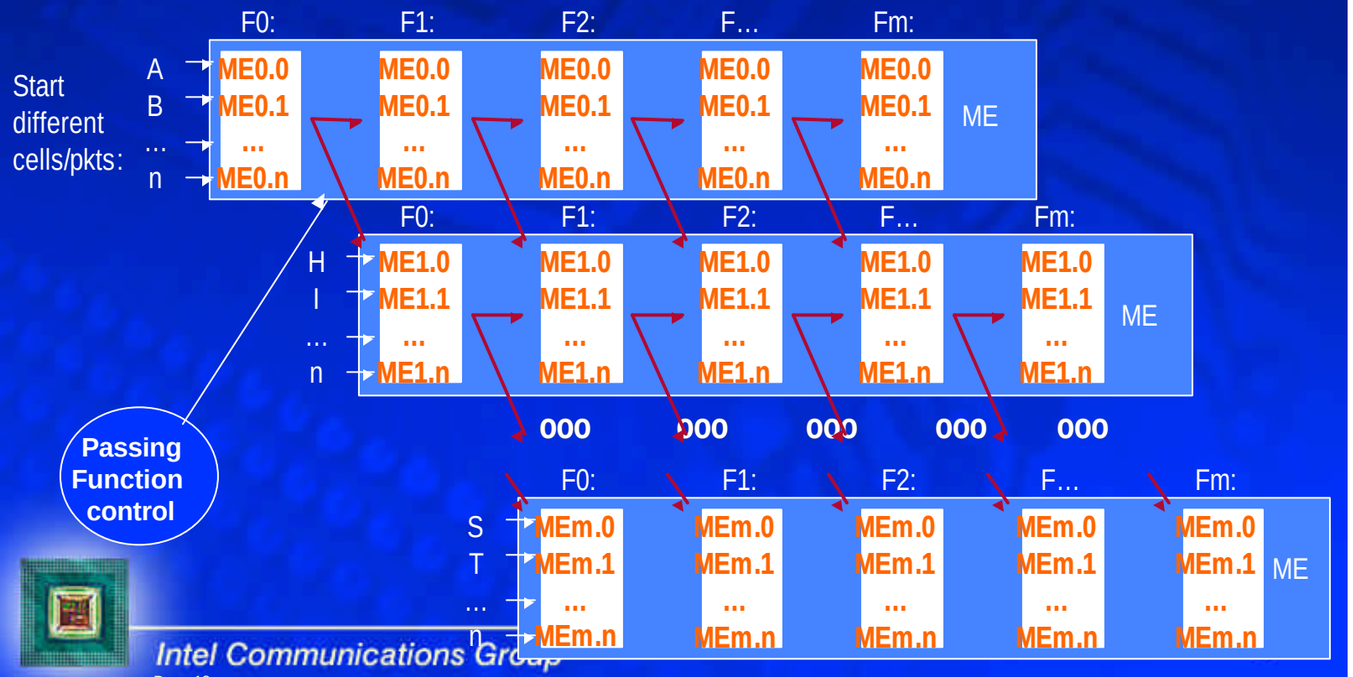
Software pipelining

- Break the problem into sequential serial chunks – pipeline stages.
- No pipe stage may be greater than the arrival rate of incoming cells or packets.
  - Two obvious approaches:
  - Make every pipe stage < arrival rate
  - Aggregate work arrival pipe stages < (arrival rate * aggregation)
- NOTE: OC-48 arrival rate = 160ns
  OC-192 arrival rate = 35ns
  OC-768 arrival rate = 8ns



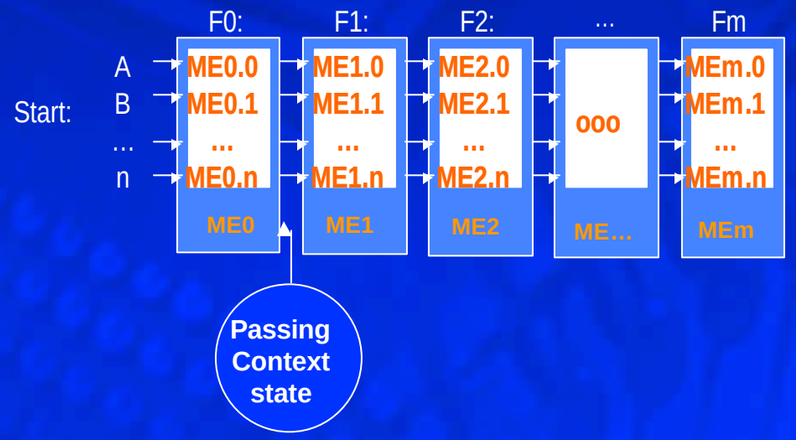
Read Modify Write (RMW) Latency Solutions:

- **Function pipelining:**
  - MEm+1 Fi waits for start semaphore from MEn Fi

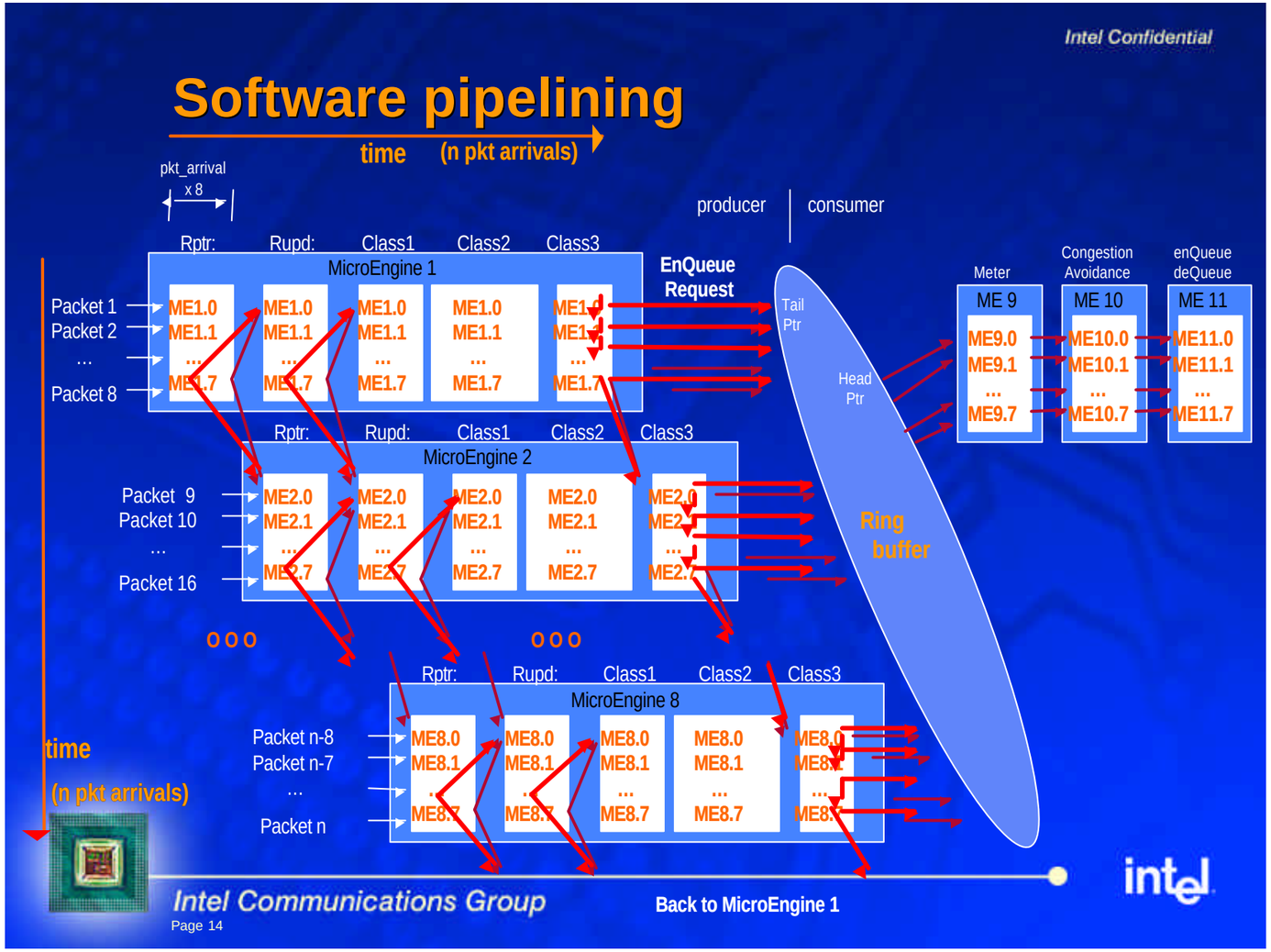


Read Modify Write (RMW) Latency Solutions:

- **Context pipelining:**
  - Each function time < (arrival rate * #threads(n))
  - Context / relevant variables must be passed from microengine MEm to MEm+1 for each cells / packet
  - Each microengine only needs software for resident function

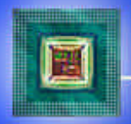


Software pipelining



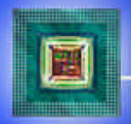
Demonstration

L3/4 forwarding at 10Gbs wire rate demonstrated using the same development environment as the IXP1200 Network Processor



Summary

- A fully programmable store and forward architecture is optimum to deliver flexible policing, QoS and advanced traffic management
- The Intel solution includes:
  - Custom .13u technology [1] ASIC .13u technology
  - Build on network processing experience, microengine version 2 optimized for software pipelining
  - Software pipelining provides performance with flexibility
  - Tools – optimized to unleash the performance, simplify programming and support portability across multiple platforms



One architecture drives multiple products